Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

ATTORNEY DOCKET NO. 10013444-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): David Jerome Johnson et al

Confirmation No.: 1931

Application No.: 10/001594

Examiner: Chace, Christian

Filing Date: Oct 31, 2001

Group Art Unit: 2187

Title: Identification Of Stale Entries In A Computer Cache

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on 11/19/2004.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $340.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

- ( ) one month $110.00
- ( ) two months $430.00
- ( ) three months $980.00
- ( ) four months $1530.00

(X) The extension fee has already been filled in this application.

( ) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account 08-2025 the sum of $340.00. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

( ) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450. Date of Deposit: ______
OR
(X) I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (703) 872-9306 on Nov. 19, 2004

Number of pages: 21

Typed Name: Donna M Kraft

Signature: Donna M Kraft

Respectfully submitted,

David Jerome Johnson et al

By A. W. Winfield

Augustus W Winfield

Attorney/Agent for Applicant(s)
Reg. No. 34,046

Date: Nov. 19, 2004

Telephone No.: 970 898 3142

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Mail Stop 35
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 10013444-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): David J.C. Johnson & Jonathan P. Lotz

Serial No.: 10/001,594 **Examiner:** Chace, Christian

Filing Date: 10/31/2001 **Group Art Unit:** 2187

Title: Identification of stale entries in a computer cache

COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria VA 22313-1450

RECEIVED
CENTRAL FAX CENTER
NOV 19 2004

BRIEF ON APPEAL

INTRODUCTION

Pursuant to the provisions of 37 CFR Part 41, Subpart B, applicants hereby appeal to the Board of Patent Appeals and Interferences from the examiner's final rejection dated 07/26/2004. A notice of appeal was timely filed on 11/19/2004, concurrently with this brief on appeal, in accordance with 37 CFR § 41.31(a)(1).

REAL PARTY IN INTEREST

The entire interest in the present application has been assigned to Hewlett-Packard Development Company, L.P., as recorded at reel 014061, frame 0492.

RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

STATUS OF CLAIMS

RECEIVED
CENTRAL FAX CENTER
NOV 1 9 2004

Claims 1-10 have been finally rejected.

Claims 1-10 are on appeal.

STATUS OF AMENDMENTS

There are no after-final amendments.

SUMMARY OF CLAIMED SUBJECT MATTER

The invention relates generally to computer systems and more specifically to cache memory systems. In an example embodiment of the invention, a cache system can identify memory lines that have not been recently used, with only a single bit per line instead of a multiple bit counter or timer per line. Figure 1 illustrates an example cache system in which the invention may be implemented. A single age-bit may provided for each line in the cache (figure 1, 114; page 6, lines 21-26), or a single age-bit may be provided for each index (figure 1, 112). The age-bits are initially set to a first logical state (figure 3, 300; page 7, lines 17-19). Each time a line, or index, is accessed (figure 2A, 200) (or alternatively, written) (figure 2B, 204), by a processor, the corresponding age-bit is reset to the first logical state (figure 2A, 202; figure 2B, 206; page 7, lines 21-25). A state machine (figure 1, 116) periodically checks the status of each age-bit (figure 3, 306; page 8, lines 3-9). If the state machine detects that an age-bit is in the first logical state, the state machine sets the age-bit to a second logical state (figure 3, 308; page 8, lines 9-11). If the state machine detects that an age-bit is already in the second logical state, then the set of lines corresponding to the index corresponding to the age-bit, or line of data corresponding to the age-bit, has not been accessed or changed since the last time the state machine checked the age-bit. If there is one age-bit per line, the line may be pre-emptively evicted (figure 3, 310; page 9, lines 15-20). If there is one age-bit per index value, a replacement algorithm may be use to determine a line to evict, for example, the least-recently-used line corresponding to the index (figure 3, 310; page 8, lines 12-25).

Claim 1 specifies a cache memory system (figure 1), comprising storage for a plurality of data values (figure 1, 106); storage for a plurality of age bits (figure 1, 112, 114), each age bit corresponding to one of the data values; and each age bit indicating whether the corresponding data value has been recently accessed (figures 2A and 2B; page 7, line 16 through page 8, page 11).

Claim 2, dependent on claim 1, further specifies: each age bit indicating that the corresponding data value has not been recently accessed when the age bit has remained at a particular logical state for at least a predetermined time period (figures 2A and 2B; page 7, line 16 through page 8, page 11).

Claim 3, dependent on claim 2, further specifies: a state machine, the state machine periodically determining the state of each age bit, and for each age bit that is not at the particular logical state, setting the state of the age bit to the particular logical state (figure 3; page 8, lines 4-27).

Claim 4, dependent on claim 1, further specifies: each age bit further indicating whether the corresponding data value is modified (figure 2B; page 7, line 23, through page 8, line 2).

Claim 5, dependent on claim 4, further specifies: each age bit indicating that the corresponding data value is modified and has not been recently accessed when the age bit has remained at a particular logical state for at least a predetermined time period (figure 2B; page 7, line 23, through page 8, line 2).

Claim 6, dependent on claim 5, further specifies: a state machine, the state machine periodically determining the state of each age bit, and for each age bit that is not at the particular logical state, setting the state of the age bit to the particular logical state (figure 3; page 8, lines 4-27).

Claim 7 specifies a method of detecting whether an entry in a cache memory has been recently accessed, comprising: setting a bit to a first logical state when the entry is accessed; setting the bit to a second logical state; and determining that the entry has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state (figure 3; page 8, lines 4-27).

Claim 8 specifies a method of detecting whether an entry in a cache memory is dirty and has not been recently accessed, comprising: setting a bit to a first logical state when the

entry is written; setting the bit to a second logical state; and determining that the entry is dirty and has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state (figure 2B and figure 3; page 8, lines 4-27).

Claim 9 specifies a method of detecting whether at least one entry in a set of entries in a cache memory has been recently accessed, comprising: setting a bit to a first logical state when an entry corresponding to an index is accessed; setting the bit to a second logical state; and determining that at least one entry corresponding to the index has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state (figure 3; page 8, lines 4-27).

Claim 10 specifies a method of detecting whether at least one entry in a set of entries in a cache memory is dirty and has not been recently accessed, comprising: setting a bit to a first logical state when an entry corresponding to an index is modified; setting the bit to a second logical state; and determining that at least one entry corresponding to the index is dirty and has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state (figure 2B and figure 3; page 8, lines 4-27).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Whether claims 1-10 are unpatentable under 35 U.S.C. § 102 as anticipated by U.S. Patent Number 5,655,103 (Cheng *et al.*)

ARGUMENT

Claim 1

Claim1 specifies age bits in a cache. Cheng *et al.* do not teach or suggest age bits in a cache.

In Cheng *et al.*, the stale bits are in a memory controller, not in a cache. Cheng *et al.*, figure 1 illustrates CPU's with caches 101, 102, and 103, and memory controller 107.

Figure 3 illustrates a dependency table 109, which includes the stale bit. From column 5, lines 10-11 and lines 27-33, the stale bits are in the dependency table in the memory controller 107, not in the caches 101, 102, and 103.

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987).

"The identical invention must be shown in as complete detail as is contained in the . . . claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).

"The elements must be arranged as required by the claim, but . . . identity of terminology is not required." *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990).

In the office action dated 07/26/2004, at page 9, the examiner argues that claim 1 specifies a cache system, which could include a memory controller. Assuming for the sake of argument that the cache system of claim 1 could include a memory controller, the memory controller and stale bits in Cheng *et al.* are not in a cache system, and accordingly in Cheng *et al.* the elements are not arranged as required by the claim.

Claim 2

Claim 2 further specifies that each age bit indicates that the corresponding data value has <u>not</u> been recently accessed when the age bit has remained at a particular logical state for at least a predetermined time period. Cheng *et al.* do not teach or suggest an age bit that indicates that corresponding data has not been recently accessed when the age bit has remained at a particular logical state for at least a predetermined time period.

In Cheng *et al.*, a stale bit is associated with an address in the dependency table. If the stale bit is at logical zero for a predetermined time, then that does not indicate that the corresponding address was not recently accessed. For example, in figures 4-5, the inclusion bits and cross interrogation bits change state three times while the stale bit remains at logical state zero, and the memory controller accesses the address in the dependency table each time it changes an inclusion bit or cross interrogation bit. Likewise, if the stale bit is

at logical one for a predetermined time, then it does not indicate that the corresponding address was not recently accessed. For example, in figures 7 and 8, the inclusion bits and stale bits change state while the stale bit remains at logical state one. The memory controller accesses the address in the dependency table each time it changes an inclusion bit or cross interrogation bit.

Claim 3

Claim 3 further specifies: a state machine, the state machine periodically determining the state of each age bit, and for each age bit that is not at the particular logical state, setting the state of the age bit to the particular logical state. Cheng *et al.* do not teach or suggest a state machine, the state machine periodically determining the state of each age bit, and for each age bit that is not at the particular logical state, setting the state of the age bit to the particular logical state.

In Cheng *et al.*, a stale bit is initially set to logical zero, and it remains at logical zero until the requesting cache casts out newly modified data to be written at the address corresponding to the stale bit. The stale bit then remains at logical one until the memory controller receives cast out data from a sending cache, to be written at the address corresponding to the stale bit. Nothing about setting a stale bit to its initial value or changing it to logical one is dependent on a state machine periodically determining the state of the stale bit.

In the office action dated 07/26/2004, at page 3, in conjunction with claim 2, the examiner equates the particular logical state of claim 2 to a stale bit at logical state zero. In addition, in conjunction with claim 3, the examiner equates the memory controller of Cheng *et al.* to a state machine, and equates the time required for a processor to modify a cache line to a time period for determining the state of each age bit. Assume for the sake of argument that the examiner is correct: in accordance with claim 3 the memory controller must wait until the requesting processor modifies a cache line, and then determine whether the age bit is logical zero, and if not, set it to logical zero. However, this is inconsistent with Cheng *et al.* In Cheng *et al.*, when the requesting processor modifies a cache line, the age bit is set to logical one. In addition, Cheng *et al.* do not teach or suggest that the controller must determine the state of the age bit before setting it to logical one.

Claim 4

Claim 4, dependent on claim 1, further specifies: each age bit further indicating whether the corresponding data value is modified. Cheng *et al.* do not teach or suggest each age bit further indicating whether the corresponding data value is modified.

From the discussion above in conjunction with claim 1, the data corresponding to a stale bit is an address. In the office action dated 07/26/2004, at page 2, in conjunction with claim 1, the examiner equates the address, inclusion bits, and cross-interrogation bits in the dependency table as the data corresponding to a stale bit. The state of the stale bit does not indicate one way or the other whether the address alone or address plus other dependency table bits have been modified.

In the office action dated 07/26/2004, at page 4, the examiner equates the cache line corresponding to the address as the data corresponding to a stale bit. However, claim 4 is dependent on claim 1, and claim 1 specifies that the age bit and the corresponding data are in the same structure (a cache). In Cheng *et al.*, the stale bits are not in the same structure as the cache line.

Claim 5

Claim 5, dependent on claim 4, further specifies: each age bit indicating that the corresponding data value is modified and has <u>not</u> been recently accessed when the age bit has remained at a particular logical state for at least a predetermined time period. Cheng *et al.* do not teach or suggest that each age bit indicates that the corresponding data value is modified and has <u>not</u> been recently accessed when the age bit has remained at a particular logical state for at least a predetermined time period.

Applicant's arguments above in conjunction with claim 2 apply equally to claim 5.

In the office action dated 07/26/2004, at page 4, for claim 5, the examiner equates a cache line as the data corresponding to a stale bit, equates a stale bit at logical state one as the particular logical state, and equates the time it takes a processor to modify a line as the predetermined time period. Assuming for the sake of argument that the examiner is correct,

in accordance with claim 5, a stale bit must indicate that a cache line has been modified and has not been recently accessed when the stale bit has remained at a logical one for the time period required for a processor to modify the cache line. This is inconsistent with Cheng *et al.* In Cheng *et al.*, the memory controller sets the stale bit to logical one after the cache line is modified, and when the stale bit is at logical one, the cache line has been recently accessed (cast out).

Claim 6

Claim 6, dependent on claim 5, further specifies: a state machine, the state machine periodically determining the state of each age bit, and for each age bit that is not at the particular logical state, setting the state of the age bit to the particular logical state. Cheng *et al.* do not teach or suggest a state machine, the state machine periodically determining the state of each age bit, and for each age bit that is not at the particular logical state, setting the state of the age bit to the particular logical state.

In the office action dated 07/26/2004, at page 3, the examiner argues claims 3 and 6 together. However, the examiner's arguments are inconsistent. Claim 3 is dependent on claim 1, for which the examiner equates the particular logical state as a stale bit set to zero. Claim 6 is dependent on claim 5, and for claim 5 the examiner has effectively equated the particular logical state as a stale bit set to one. Assuming for the sake of argument that the particular logical state is a stale bit set to one, then in accordance with claim 6, the memory controller must wait for the time required to modify a line, determine the state of each stale bit, and for each stale bit that is not a logical state one, set the state of the stale bit to logical state one. The applicant submits that Cheng *et al.* do not teach or suggest that the memory controller must determine the state of the stale bit before setting the stale bit. If the requesting processor's cache casts out a line, the memory controller can simply set the stale bit without determining its previous state.

Claim 7

Claim 7 specifies a method of detecting whether an entry in a cache memory has been recently accessed, comprising: setting a bit to a first logical state when the entry is accessed; setting the bit to a second logical state; and determining that the entry has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state. Cheng *et al.* do not teach or suggest a method of detecting whether an entry in a cache memory has been recently accessed, comprising: setting a bit to a first logical state when the entry is accessed; setting the bit to a second logical state; and determining that the entry has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state.

In the office action dated 07/26/2004, at page 4 in conjunction with claim 7, the examiner submits that in Cheng *et al.*, a stale bit is set to a logical state each time an entry is accessed. Applicant disagrees. In Cheng *et al.*, a stale bit is created only if there is a load miss, and "Accessed" includes more than a load miss. Stated alternatively, if the entry is in the cache, then there is no load miss. The preamble of claim 7 specifies detecting whether <u>an entry in a cache memory</u> has been recently accessed, and access of an entry in a cache memory does not generate a load miss.

In addition, claim 7 specifies setting a bit to a first logical state when the entry is accessed; setting the bit to a second logical state; and determining that the entry has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state.

In the office action dated 07/26/2004, at page 5, the examiner equates the first logical state as a stale bit set to logical zero, the second logical state as a stale bit set to logical one, and the predetermined time as the time required for a processor to modify a cache line. However, as discussed in conjunction with claim 5, this is inconsistent with Cheng *et al.* In Cheng *et al.*, the memory controller sets the stale bit to logical one <u>after</u> the cache line is modified. There is no predetermined time after the stale bit has been set to logical one that indicates that a cache entry has not been accessed.

Claim 8

Claim 8 specifies a method of detecting whether an entry in a cache memory is dirty and has not been recently accessed, comprising: setting a bit to a first logical state when the entry is written; setting the bit to a second logical state; and determining that the entry is dirty and has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state. Cheng *et al.* do not teach or suggest a method of detecting whether an entry in a cache memory is dirty and has not been recently accessed, comprising: setting a bit to a first logical state when the entry is written; setting the bit to a second logical state; and determining that the entry is dirty and has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state.

Applicant's arguments in conjunction with claim 7 apply equally to claim 8, and in addition, claim 8 specifies determining that the entry is dirty when the bit is at the second logical state after at least a predetermined time after being set to the second logical state. In Cheng *et al.*, the stale bit is set to logical one after the requesting cache further modifies an already dirty line, and there is no predetermined time <u>after</u> the stale bit has been set to logical one that indicates that a cache entry is dirty and has not been recently accessed.

Claim 9

Claim 9 specifies a method of detecting whether at least one entry in a set of entries in a cache memory has been recently accessed, comprising: setting a bit to a first logical state when an entry corresponding to an index is accessed; setting the bit to a second logical state; and determining that at least one entry corresponding to the index has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state. Cheng *et al.* do not teach or suggest a method of detecting whether at least one entry in a set of entries in a cache memory has been recently accessed, comprising: setting a bit to a first logical state when

an entry corresponding to an index is accessed; setting the bit to a second logical state; and determining that at least one entry corresponding to the index has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state.

Applicant's arguments above in conjunction with claim 7 apply equally to claim 9. In addition, claim 9 adds an index and entries corresponding to the index. In Cheng *et al.*, a stale bit corresponds to one address, not to an index where there is a set of cache entries corresponding to the index. In the office action dated 07/26/2004, at page 6, the examiner equates the dependency table of Cheng *et al.* to an index, and at page 7 equates a stale bit set to one as the second logical state, and the predetermined time as the time required for a processor to modify a line. Assuming for the sake of argument that the dependency table is an index, that a stale bit set to one is the second logical state, and that the predetermined time is the time required for a processor to modify a line, then in accordance with claim 9, a stale bit must be set to logical zero when an entry corresponding to the dependency table is accessed, setting the stale bit to logical one; and determining that at least one entry corresponding to the dependency table has not been recently accessed when the stale bit is at logical one after at least the time required for a processor to modify a line after being set to logical one. In addition to the logical inconsistencies noted in conjunction with claim 7, the notion of a cache entry corresponding to the dependency table makes no sense technically.

Claim 10

Claim 10 specifies a method of detecting whether at least one entry in a set of entries in a cache memory is dirty and has not been recently accessed, comprising: setting a bit to a first logical state when an entry corresponding to an index is modified; setting the bit to a second logical state; and determining that at least one entry corresponding to the index is dirty and has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state. Cheng *et al.* do not teach or suggest a method of detecting whether at least one entry in a set of entries in a cache memory is dirty and has not been recently accessed, comprising: setting a bit to

a first logical state when an entry corresponding to an index is modified; setting the bit to a second logical state; and determining that at least one entry corresponding to the index is dirty and has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state.

Applicant's remarks in conjunction with claim 8, plus applicant's remarks in conjunction with claim 9, apply equally to claim 10.

CONCLUSION

In view of the above, applicant respectfully requests that the examiner's rejection of claims 1-10 be reversed.

Respectfully submitted,

A. W. Winfield

Augustus W. Winfield
Reg. No. 34,046

November 18, 2004
Fort Collins, CO 80528-9599
(970) 898-3142

APPENDIX 1

CLAIMS ON APPEAL

1. A cache memory system, comprising:

   storage for a plurality of data values;

   storage for a plurality of age bits, each age bit corresponding to one of the data values; and

   each age bit indicating whether the corresponding data value has been recently accessed.

2. The cache memory system of claim 1, further comprising:

   each age bit indicating that the corresponding data value has not been recently accessed when the age bit has remained at a particular logical state for at least a predetermined time period.

3. The cache memory system of claim 2, further comprising:

   a state machine, the state machine periodically determining the state of each age bit, and for each age bit that is not at the particular logical state, setting the state of the age bit to the particular logical state.

4. The cache memory system of claim 1, further comprising:

   each age bit further indicating whether the corresponding data value is modified.

5. The cache memory system of claim 4, further comprising:

   each age bit indicating that the corresponding data value is modified and has not been recently accessed when the age bit has remained at a particular logical state for at least a predetermined time period.

6. The cache memory system of claim 5, further comprising:

a state machine, the state machine periodically determining the state of each age bit, and for each age bit that is not at the particular logical state, setting the state of the age bit to the particular logical state.

7. A method of detecting whether an entry in a cache memory has been recently accessed, comprising:

setting a bit to a first logical state when the entry is accessed;

setting the bit to a second logical state; and

determining that the entry has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state.

8. A method of detecting whether an entry in a cache memory is dirty and has not been recently accessed, comprising:

setting a bit to a first logical state when the entry is written;

setting the bit to a second logical state; and

determining that the entry is dirty and has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state.

9. A method of detecting whether at least one entry in a set of entries in a cache memory has been recently accessed, comprising:

setting a bit to a first logical state when an entry corresponding to an index is accessed;

setting the bit to a second logical state; and

determining that at least one entry corresponding to the index has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state.

10. A method of detecting whether at least one entry in a set of entries in a cache memory is dirty and has not been recently accessed, comprising:

setting a bit to a first logical state when an entry corresponding to an index is modified;

setting the bit to a second logical state; and

determining that at least one entry corresponding to the index is dirty and has not been recently accessed when the bit is at the second logical state after at least a predetermined time after being set to the second logical state.